Homework 3 Grading Environments

SWPP 2021

Docker container as a grading environment

- We will provide the testing environment where your codes will be graded since each student has a different development environment (e.g., OS).
- Especially, a *Docker* container will be used for grading.
- We suggest you go through the following slides so that TAs can run your codes properly in the same environment.

What is Docker? Why Docker?

- *Docker* provides an isolated environment, called a container, for each application.
- Docker enables you to separate your applications from your infrastructure.
- So, even when you are using *Windows*, you can run your program on any other environments (e.g., *Ubuntu, Alpine* …).
- For us, we can share the environment through the container; a container with *node(v14.17.6) on Linux* will be used.
  - https://hub.docker.com/_/node

What is Docker? Why Docker?

NOTE: This material covers only the minimum requirements for checking the assignments.

If you are interested, you can check more on the details in the following links:

- **Introductions**
  - English: Introduction to Docker containers
  - Korean: Docker 컨테이너 소개
- **Practice**
  - English: Build a containerized web application with Docker
  - Korean: Docker를 사용하여 컨테이너화된 웹 애플리케이션 빌드

Install Docker (Ubuntu)

```
$ sudo apt-get update
$ sudo apt-get install apt-transport-https ca-certificates curl
software-properties-common
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
$ sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) \
   stable"
$ sudo apt-get update
$ sudo apt-get install docker-ce
```

Recommended version for Docker Engine: 20.10.XX

Install Docker

```
$ sudo docker version
```

```
# output:
Client: Docker Engine - Community
 Version:              20.10.05
 API version:          1.41
 Go version:           go1.13.15
 Git commit:           55c4c88
 Built:                Tue Mar  2 20:13:00 2021
 OS/Arch:              darwin/amd64
 Experimental:         true

Server: Docker Engine - Community
 Engine:
  Version:             20.10.05
  API version:         1.41 (minimum version 1.12)
  Go version:          go1.13.15
  ...
```

Install Docker

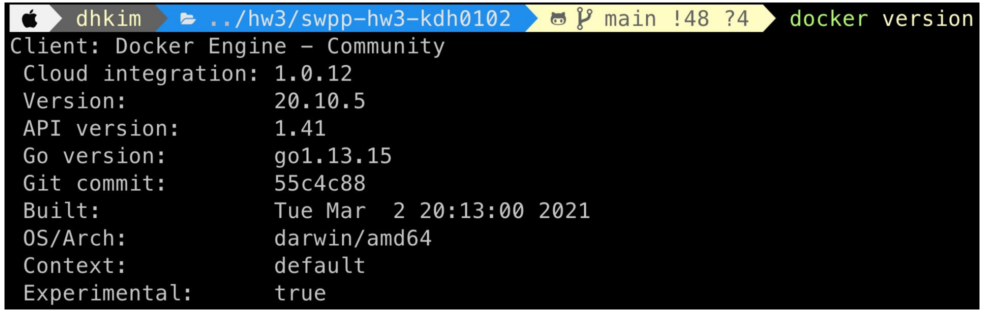
For Mac: https://docs.docker.com/desktop/mac/install/
For Windows: https://docs.docker.com/desktop/windows/install/

Let’s deploy our homework using Docker

0. Check if *Docker* is installed

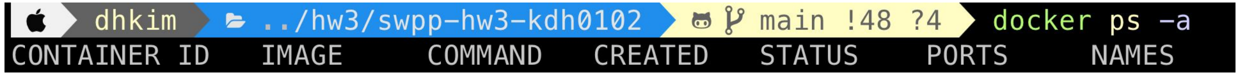
- Both the following two commands should show the proper messages:

```
$ docker version
```



```
 dhkim  ../hw3/swpp-hw3-kdh0102  main !48 ?4  docker version
Client: Docker Engine - Community
 Cloud integration: 1.0.12
 Version:           20.10.5
 API version:       1.41
 Go version:        go1.13.15
 Git commit:        55c4c88
 Built:             Tue Mar  2 20:13:00 2021
 OS/Arch:           darwin/amd64
 Context:           default
 Experimental:      true
```

```
$ docker ps -a
```

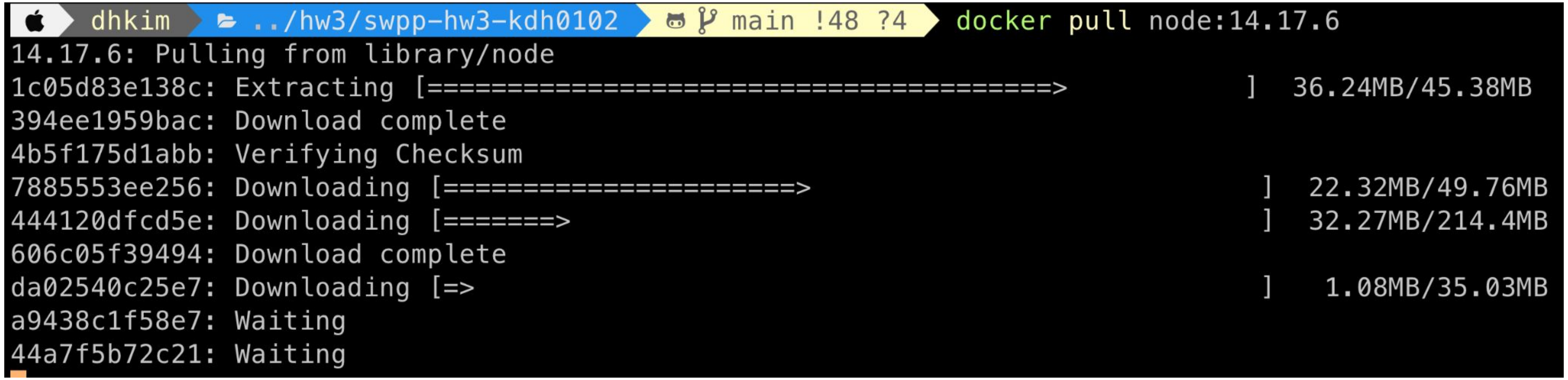


```
 dhkim  ../hw3/swpp-hw3-kdh0102  main !48 ?4  docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

1. Pull *Docker* image

- *Docker* images act as a set of instructions to build a Docker container, like a template.
- We will use the image provided by *node*, where the tag is `14.17.6`.
- Run the following commands to pull the image:

```
$ docker pull node:14.17.6
```



```
 dhkim  ../hw3/swpp-hw3-kdh0102  main !48 ?4  docker pull node:14.17.6
14.17.6: Pulling from library/node
1c05d83e138c: Extracting [========================================>          ]  36.24MB/45.38MB
394ee1959bac: Download complete
4b5f175d1abb: Verifying Checksum
7885553ee256: Downloading [======================>                            ]  22.32MB/49.76MB
444120dfcd5e: Downloading [=======>                                           ]  32.27MB/214.4MB
606c05f39494: Download complete
da02540c25e7: Downloading [=>                                                 ]   1.08MB/35.03MB
a9438c1f58e7: Waiting
44a7f5b72c21: Waiting
```

2. Run *Docker* container

- Run a *docker* container with the downloaded image.
- Use `docker run` command as follows:

```
$ docker run -it \
-p 127.0.0.1:3000:3000 \
-v ${PWD}:/app \
--name hw3 \
node:14.17.6 \
/bin/bash
```

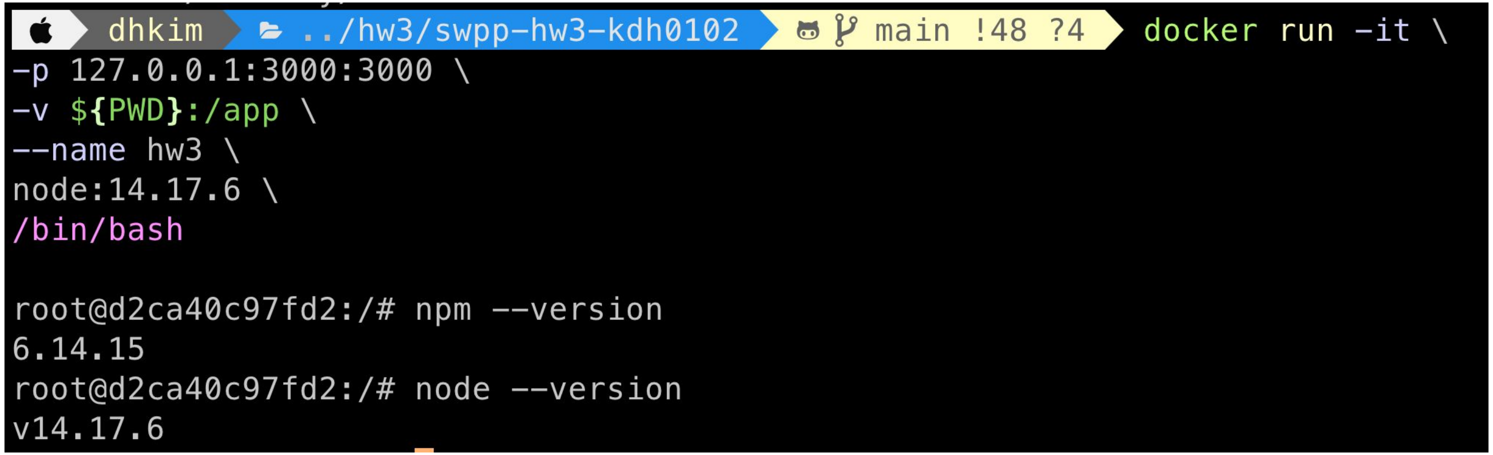
Bind host port 3000 to container port 3000.

Mount the current directory to the container path (*/app*). **Note the current directory is the root of your homework repo.**

Container name.

2. Run *Docker* container

- Run a *docker* container with the downloaded image.
- Use docker run command as follows:
- Now you are inside the container!



```
 dhkim  ../hw3/swpp-hw3-kdh0102  main !48 ?4  docker run -it \
-p 127.0.0.1:3000:3000 \
-v ${PWD}:/app \
--name hw3 \
node:14.17.6 \
/bin/bash

root@d2ca40c97fd2:/# npm --version
6.14.15
root@d2ca40c97fd2:/# node --version
v14.17.6
```

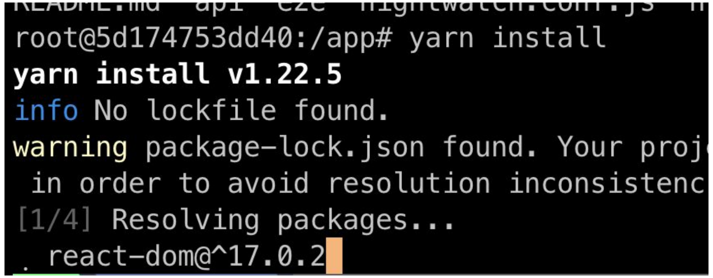
3. Install packages

- Let’s install the required packages.
- First of all, move to your directory (/app).

```
$ cd /app
```

- Then, install packages. Note that the results (node_modules/) will be saved in your host (i.e., outside of the container) since we mounted the path.

```
$ yarn install
```

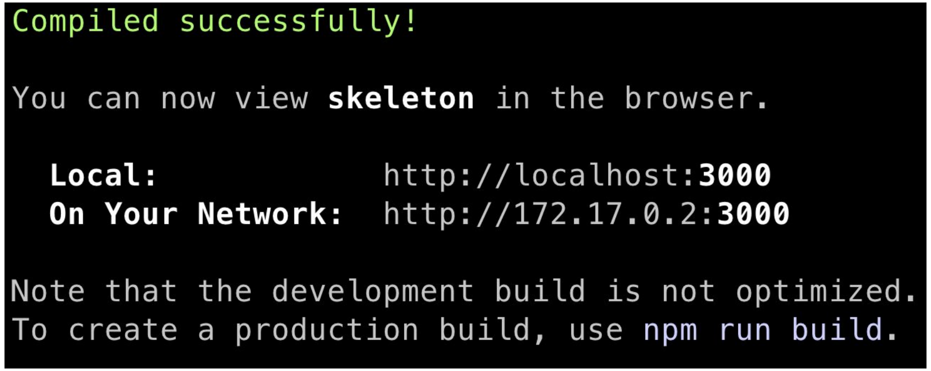


```
root@5d174753dd40:/app# yarn install
yarn install v1.22.5
info No lockfile found.
warning package-lock.json found. Your proj
 in order to avoid resolution inconsistenc
[1/4] Resolving packages...
  react-dom@^17.0.2
```

4. Deploy the server

- Now you are ready to deploy your assignment!
- Let’s run backend and start together.

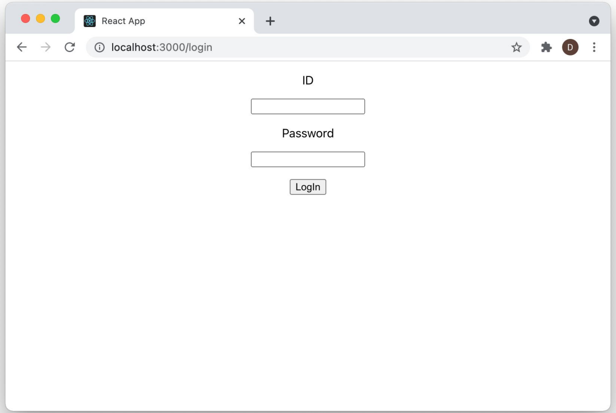
```
$ npm run backend & yarn start
```



```
Compiled successfully!

You can now view skeleton in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://172.17.0.2:3000

Note that the development build is not optimized.
To create a production build, use npm run build.
```

4. Deploy the server

- Now you are ready to deploy your assignment!
- Let’s run `backend` and `start` together.
- In your browser, try access `localhost:3000` and check if all is working properly as in your own environments.
- FYI, the grading will be conducted with chromedriver@93.0.1.


React App
localhost:3000/login
ID
Password
LogIn